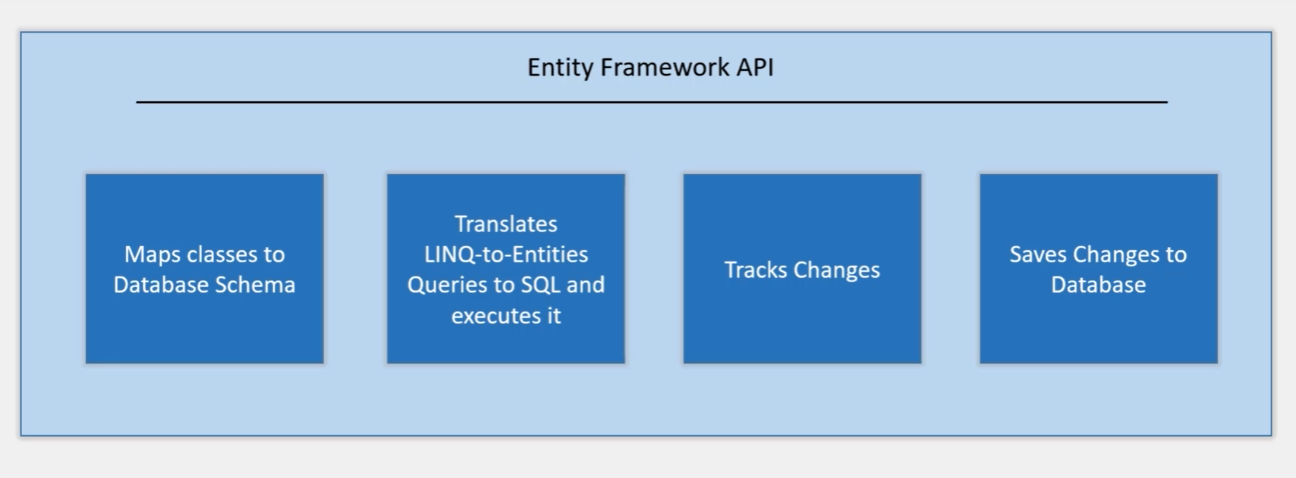
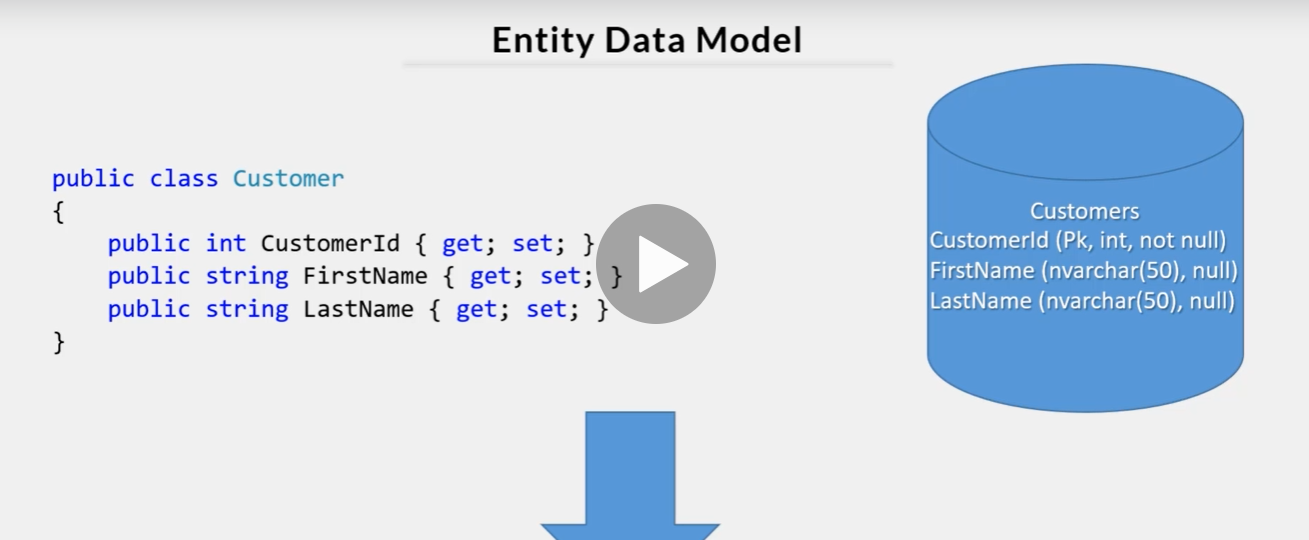
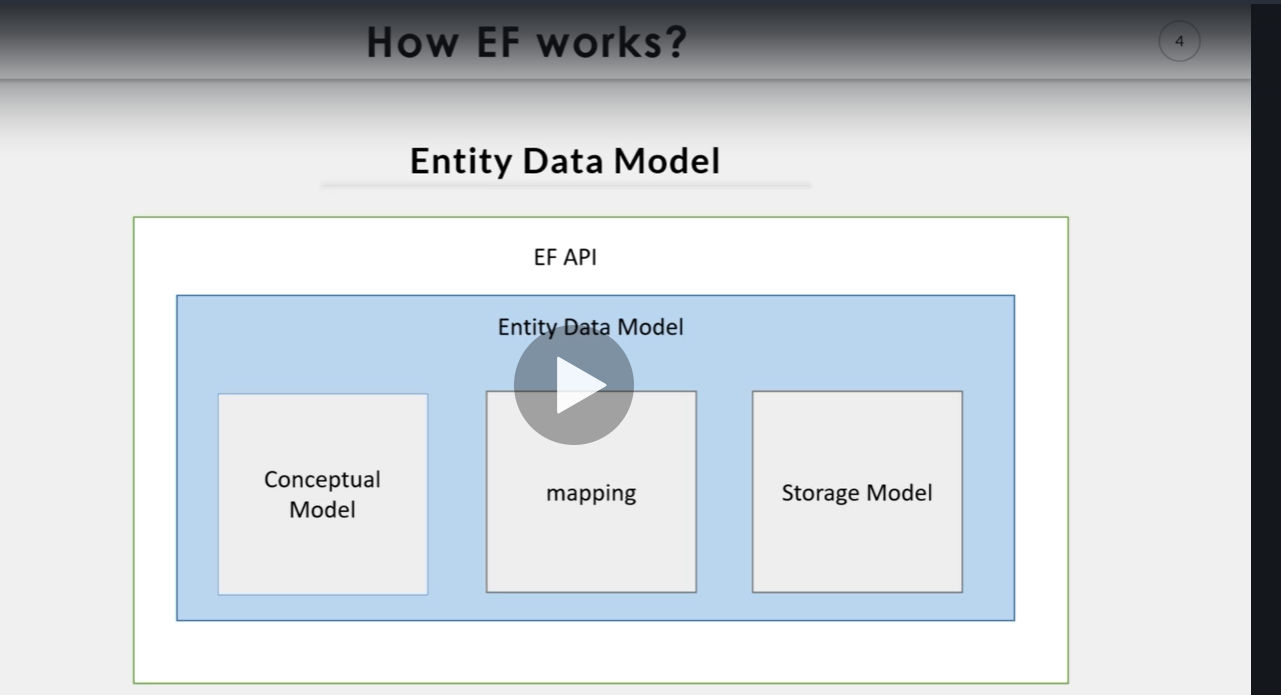
**How EF works…**





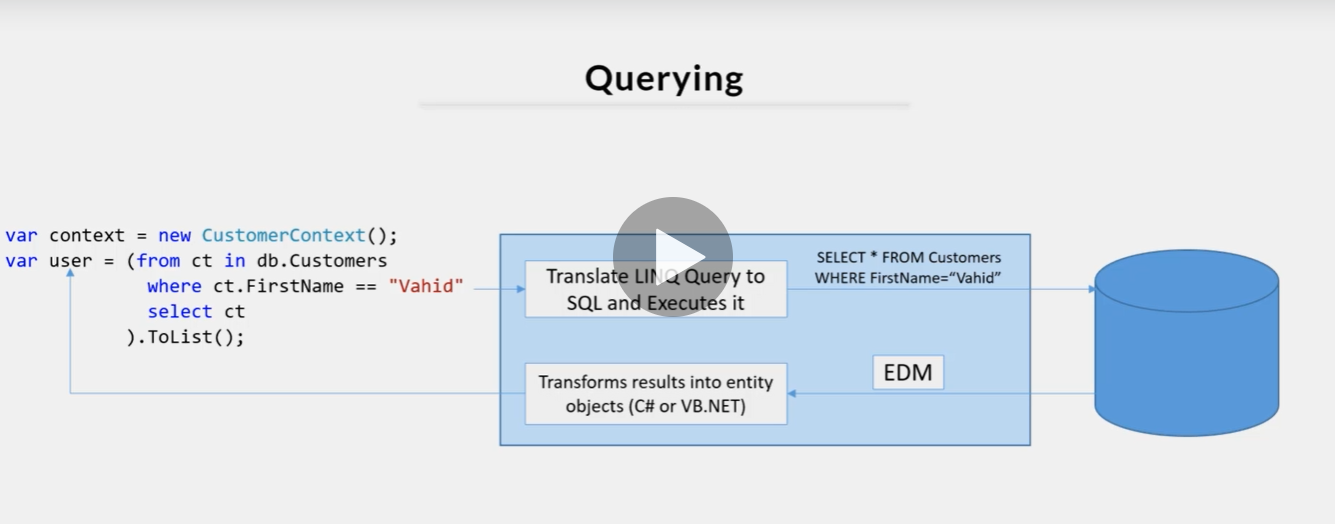
Above is the data model. In this case customer is the class and we can see properties of it within it.



1. Its first task is to build data model i.e. EDM(Entity data model)
2. Its in-memory representation of entire metadata. (conceptual model , storage model and mapping between them)
3. It builds conceptual model from domain class, context class
4. Default conversion is followed in your domain class and configuration.
5. EF builds storage model for underlying database schema.
6. Code first approach will be inferred from conceptual model. Whereas database first approach will be inferred from database.
7. Mapping refers to how conceptual model maps to database schema i.e. storage model.

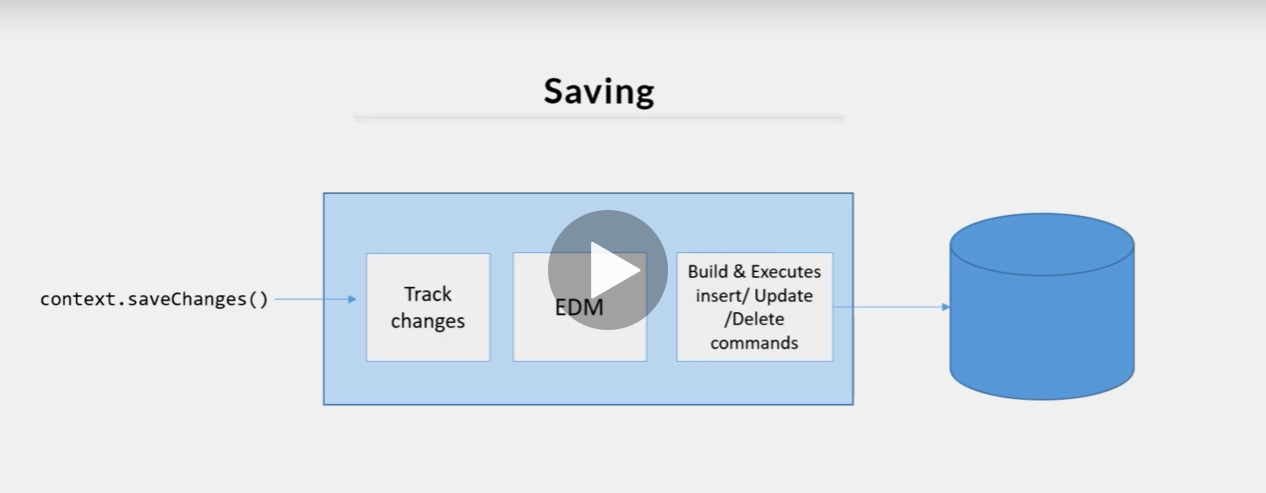
**Querying…**

1. EF perform CRUD operations using EDM
2. Its uses EDM for building SQL queries from LINQ queries…building INSERT, DELETE, UPDATE commands, and transform database results into ENTITY object.



So here first LINQ query is converted into SQL query for querying EDM. EDM returns the ENTITY object i.e. user in above case.

**Saving…**

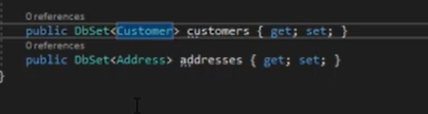


EF API infers insert, update, delete command based on state of entities whenever save changes is called. Change track keeps track of each entities as and when action is performed.

**What is DbContext ?**

* Shows session with underlying database by which can perform CRUD(Create Read Update Delete) operations.
* It is a class derived from System.Data.Entity.DbContext in entity framework core and entity framework 6 both
* Used to query and save data to database.
* It is also used to configure domain classes, data base related mappings, Caching, change tracking settings, Transactions & etc.

**Entity in Entity Framework**



Suppose we have one project called CustomerExample which prominently have customer.cs and order.cs and one class called customerDbContext.cs

customerDbContext class inherits from DbContext class(install EF framework and import it into project).

Customer & Order are domain classes in project CustomerExample.

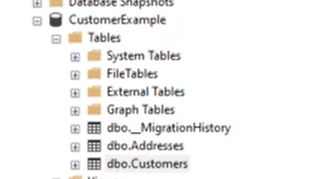
These classes becomes entities when they are included as **Dbset<TEntity>** property in context class, which is the class that is derived from DbContext class.

Now in customerDbContext we need to define above two properties in way like **DbSet<TEntity> property\_name { get ; set ; }**

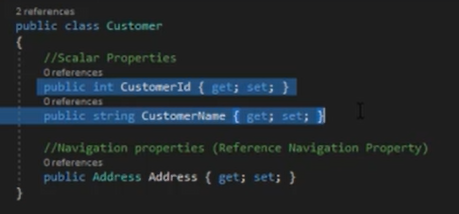
In above pic Customer and Address are **entities** Whereas

customers & addresses are called **entity sets**

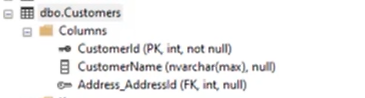
Entity Framework API will map these **entities** to **database tables**. And **Entities** properties will map to **column of database**.



Scalar VS Navigational properties in Entity

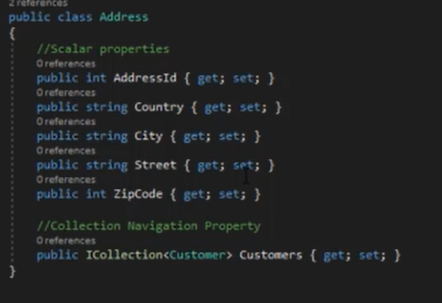


The primitive type properties are called scalar properties.(eg CustomerId, CustomerName). These properties the refer to columns of the table.



Navigational Properties define relation between two different properties. There are two types of navigational properties viz reference navigation & collection navigation.

1. If entity includes reference to another entity type is called reference navigation property. In above pic you can see that Address\_AddressId is added as foreign key in customers table. Represent 1 to 1 relationship.
2. If entity contains property of type generic collection of an entity type . It is called as collection navigation property. is called as multiplicity of many.



EF API does not create any column for collection navigation property in related table of an entity. It rather creates columns in table of an entity of generic collection.

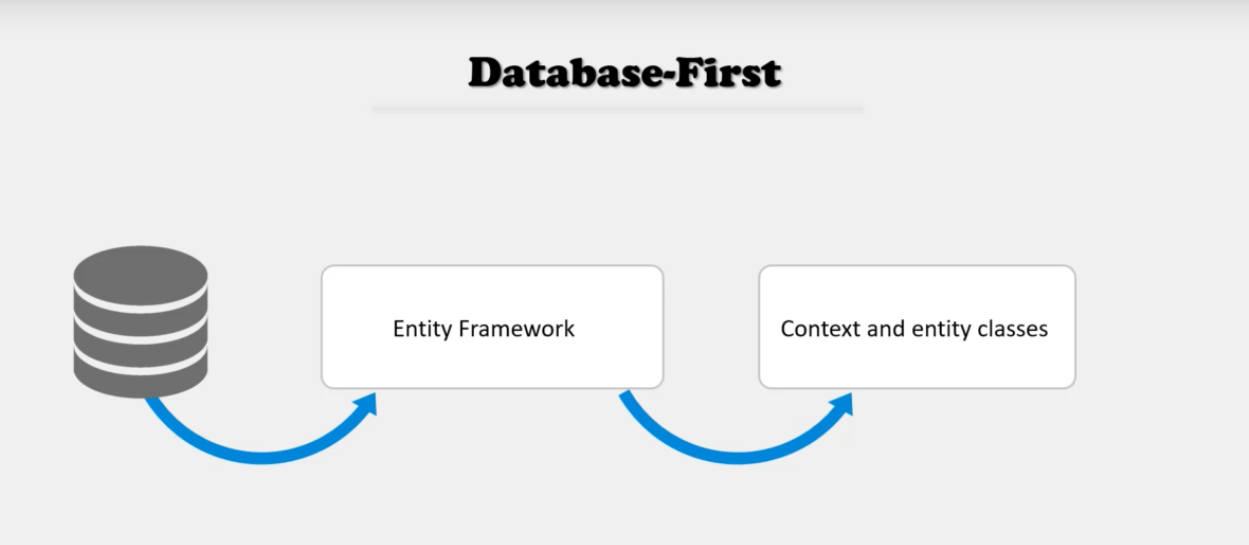
In above 2nd point we can see image that contains Address class. So, the Addresss entity has one generic collection navigation property Icollection<customer>. Customer entity is specified as generic type. Hence EF will column create Address\_AddressID in customer table in the database.

**Entity Framework Approaches**

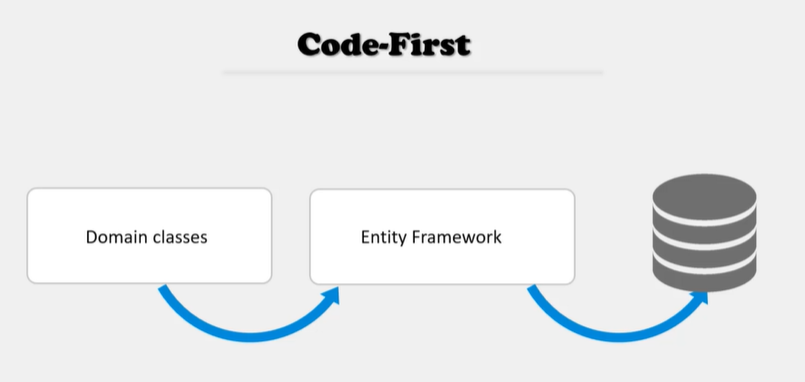
Before you develop your application you need to create your database in EF.

Types Of Development Approaches with Entity Framework.

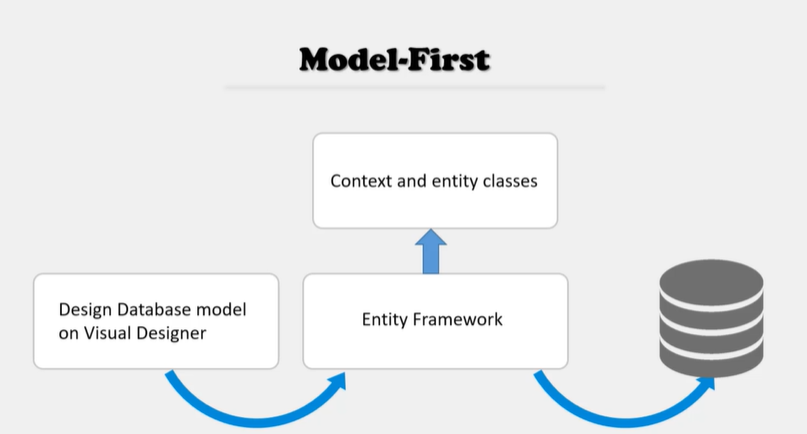
1. Database-First



1. Code -First



1. Model-First



When to use what

